

# Resolución Primer Parcial

## Ej. 1. Especificación: axiomatización

```
1  axiomas:   $\forall s, t: \text{secuComp}(\alpha), \forall s': \text{secu}(\alpha), \forall e: \text{tupla}(\alpha, \text{nat})$ 
2    vacía?: secuComp( $\alpha$ ) s -> bool
3    vacía?(<>)   $\equiv$  true
4    vacía?(e•s)  $\equiv$  false
5
6    prim:  secuComp( $\alpha$ ) s -> tupla( $\alpha$ , nat)  {¬vacía?(s)}
7    prim(e•s)   $\equiv$  if (¬vacía?(s))  $\wedge$   $\pi_1(\text{prim}(s)) = \pi_1(e)$  then
8                  < $\pi_1(e), \pi_2(e) + \pi_2(\text{prim}(s))$ >
9                  else
10                   e
11                  fi
12    fin:  secuComp( $\alpha$ ) s -> secuComp( $\alpha$ )  {¬vacía?(s)}
13    fin(e•s)   $\equiv$  if (¬vacía?(s))  $\wedge$   $\pi_1(e) = \pi_1(\text{prim}(s))$  then
14                  fin(s)
15                  else
16                   s
17                  fi
18  //-----
19    comprimir: secu( $\alpha$ ) -> secuComp( $\alpha$ )
20    comprimir(s')   $\equiv$  if vacía?(s') then
21                      <>
22                      else
23                      <prim(s'), 1> • comprimir(fin(s'))
24                      fi
25  //-----
26    maxApsContiguas:  secuComp( $\alpha$ ) ->  $\alpha$   {¬vacía?(s)}
27    maxApsContiguas(s)   $\equiv$   $\pi_1(\text{tuplaConMayor}_\pi_2(s))$ 
28
29    tuplaConMayor_ $\pi_2$ :  secuComp( $\alpha$ ) -> tupla( $\alpha$ , nat)  {¬vacía?(s)}
30    tuplaConMayor_ $\pi_2$ (s)  $\equiv$ 
31      if vacía?(fin(s))  $\vee$   $\pi_2(\text{prim}(s)) > \pi_2(\text{tuplaConMayor}_\pi_2(\text{fin}(s)))$  then
32        prim(s)
33      else
34        tuplaConMayor_ $\pi_2$ (fin(s))
35      fi
36  //-----
37    longComprimida:  secuComp( $\alpha$ ) -> nat
38    longComprimida(s)   $\equiv$  if vacía?(s) then
39                      0
40                      else
41                      1 + longComprimida(fin(s))
42                      fi
43  Fin TAD
```

## Ej. 2. Especificación: modelado

```
1 |TAD Ciudad es String
2 |TAD Persona es Tupla<Nat dni, Ciudad ciudad>
3 |TAD Sistema de Vacunación
4 |  Igualdad Observacional: ( $\forall s_0, s_1: sv$ )
5 |    ( $s_0 =_{obs} s_1 \iff$ 
6 |      ( $ciudades(s_0) =_{obs} ciudades(s_1) \wedge L$ 
7 |        ( $\forall c: ciudad$ ) ( $c \in ciudades(s_0) \Rightarrow L$ 
8 |          ( $esperando(c, s_0) =_{obs} esperando(c, s_1) \wedge$ 
9 |            ( $vacunados(c, s_0) =_{obs} vacunados(c, s_1) \wedge$ 
10 |              ( $\#frascosXCiudad(c, s_0) =_{obs} \#frascosXCiudad(c, s_1)$ 
11 |                )
12 |              )
13 |            )
14 |          )
15 |        );
16 |
17 | Géneros: sv
18 | Exporta: sv, observadores, generadores,
19 | Usa: Bool, Nat
20 | Observadores Básicos:
21 |   ciudades:      sv s      -> conj(ciudad)
22 |   esperando:     ciudad c x sv s -> conj(persona) {c ∈ ciudades(s)}
23 |   vacunados:     ciudad c x sv s -> conj(persona) {c ∈ ciudades(s)}
24 |   #frascosXCiudad: ciudad c x sv s -> nat {c ∈ ciudades(s)}
25 |
26 | Generadores:
27 |   inicioSV:      conj(ciudad) cc      -> sv {¬(∅?(cc)) }
28 |   traerfrascosDeVacuna: nat v x ciudad c x sv s -> sv {c ∈ ciudades(s) ∧ n > 0}
29 |   inscribirPersona: conj(persona) cp x ciudad c x sv s -> sv
30 |     { c ∈ ciudades(s) ∧ L (∀p: persona) ( p ∈ cp ⇒ L
31 |       (c = p.ciudad ∧ ¬(p ∈ esperando(c, s)) ∧ ¬(p ∈ vacunados(c, s))) ) }
32 |
33 | Otras Operaciones:
34 |   ciudadQueMasVacuno: sv s -> ciudad
35 |   ciudadConMayorCantFrascos: sv s -> ciudad
36 |
37 | // AUXILIARES
38 | // Recibe la cantidad de vacunas actual y el conj de personas disponibles para vacunar
39 | // y devuelve las personas vacunadas que se deben vacunar de tales parametros
40 | // De está forma modularizo la acción de vacunar cuando llegan vacunas o personas
41 | seDebenVacunar: nat v x conj(persona) cp -> conj(persona)
42 |
43 | dameN:      conj(nat) c x nat n -> conj(nat)
44 | maxEnConj:   conj(nat)      -> nat
45 |
46 | #vacunadosEnCadaCiudad: conj(ciudad) cc x sv s -> conj(nat) {-vacia?(cc) ∧ cc ⊆ ciudades(s)}
47 | #frascosEnCadaCiudad:   conj(ciudad) cc x sv s -> conj(nat) {-vacia?(cc) ∧ cc ⊆ ciudades(s)}
```

```

45 Axiomas:
46 ciudades(inicioSV(cc)) ≡ cc
47 ciudades(traerfrascosDeVacuna(v,c,s)) ≡ ciudades(v,s)
48 ciudades(inscribirPersona(cp,c,s)) ≡ ciudades(v,s)
49 //-----
50 esperando(c, inicioSV(cc)) ≡ ∅
51 esperando(c, traerfrascosDeVacuna(v,c',s)) ≡
52     if c = c' then
53         esperando(c,s) - seDebenVacunar(#frascosXCiudad(c,s) + v, esperando(c,s))
54     else
55         esperando(c,s)
56     fi
57 esperando(c, inscribirPersona(cp,c',s)) ≡
58     if c = c' then
59         esperando(c,s) - seDebenVacunar(#frascosXCiudad(c,s), esperando(c,s) u cp)
60     else
61         esperando(c,s)
62     fi
63 //-----
64 vacunados(c, inicioSV(cc)) ≡ ∅
65 vacunados(c, traerfrascosDeVacuna(v,c',s)) ≡
66     if c = c' then
67         vacunados(c,s) u seDebenVacunar(#frascosXCiudad(c,s) + v, vacunados(c,s))
68     else
69         vacunados(c,s)
70     fi
71 vacunados(c, inscribirPersona(cp,c',s)) ≡
72     if c = c' then
73         vacunados(c,s) u seDebenVacunar(#frascosXCiudad(c,s), vacunados(c,s) u cp)
74     else
75         vacunados(c,s)
76     fi
77 //-----
78 #frascosXCiudad(c, inicioSV(cc)) ≡ 0
79 #frascosXCiudad(c, traerfrascosDeVacuna(v,c',s)) ≡
80     if c = c' then
81         #frascosXCiudad(c,s) - (#( seDebenVacunar(#frascosXCiudad(c,s) + n, esperando(c,s)) )/5)
82     else
83         #frascosXCiudad(c,s)
84     fi
85 #frascosXCiudad(c, inscribirPersona(cp,c',s)) ≡
86     if c = c' then
87         #frascosXCiudad(c,s) - (#( seDebenVacunar(#frascosXCiudad(c,s), esperando(c,s) u cp) )/5)
88     else
89         #frascosXCiudad(c,s)
90     fi

```

```

91 //-----
92 seDebenVacunar: nat v × conj(persona) cp    -> conj(persona)
93 seDebenVacunar(v, cp)  ≡  if (v = 0) v (#(cp)<5) then
94     ∅
95     else
96         dameN(cp, 5) u seDebenVacunar(f-1, cp - dameN(cp, 5))
97     fi
98 //-----
99 dameN: conj(nat) c × nat n    -> conj(nat)
100 dameN(c, n) ≡ if n > 0 then Ag(dameUno(c), dameN(sinUno(c), n-1)) else <> fi
101
102 maxEnConj: conj(nat)    -> nat
103 maxEnConj(c) ≡ if ∅?(c) then 0 else máx( dameUno(c), maxEnConj(fin(c)) ) fi
104
105 //-----
106 ciudadQueMasVacuno(s)      ≡  maxEnConj(#vacunadosEnCadaCiudad(ciudades(s), s))
107
108 ciudadConMayorCantFrascos(s)  ≡  maxEnConj(#frascosEnCadaCiudad(ciudades(s), s))
109
110 #vacunadosEnCadaCiudad: conj(ciudad) cc × sv s -> conj(nat)    {¬vacía?(cc) ∧ cc ⊂ ciudades(s)}
111 #vacunadosEnCadaCiudad(cc,s) ≡
112     if ∅?(cc) then
113         <>
114     else
115         Ag( #( vacunados(dameUno(cc), s) ), #vacunadosEnCadaCiudad(sinUno(cc), s) )
116     fi
117
118 #frascosEnCadaCiudad: conj(ciudad) cc × sv s -> conj(nat)    {¬vacía?(cc) ∧ cc ⊂ ciudades(s)}
119 #frascosEnCadaCiudad(cc,s) ≡
120     if ∅?(cc) then
121         <>
122     else
123         Ag( #frascosXCiudad(dameUno(cc), s), #frascosEnCadaCiudad(sinUno(cc), s) )
124     fi
125
126 Fin TAD

```